

# Detection of Obfuscated Attacks in Collaborative Recommender Systems<sup>1</sup>

Chad Williams and Bamshad Mobasher and Robin Burke and Jeff Sandvig and Runa Bhaumik<sup>2</sup>

**Abstract.** The vulnerability of collaborative recommender systems has been well established; particularly to reverse-engineered attacks designed to bias the system in an attacker’s favor. Recent research has begun to examine detection schemes to recognize and defeat the effects of known attack models. In this paper we propose several techniques an attacker might use to modify an attack to avoid detection, and show that these obfuscated versions can be nearly as effective as the reverse-engineered models yet harder to detect. We explore empirically the impact of these obfuscated attacks against systems with and without detection, and discuss alternate approaches to reducing the effectiveness of such attacks.

## 1 Introduction

Recent work has exposed significant vulnerabilities in collaborative filtering recommender systems to what have been termed “shilling” or “profile injection” attacks in which a malicious user enters biased profiles in order to influence the system’s behavior [3, 1, 7, 10]. While there are ways system owners can increase the cost of attack profiles being created; doing so often comes at the cost of reduced participation, which can hamper the predictive accuracy of a collaborative system. As a result it is impossible to completely eliminate the threat of an attack in an open collaborative system.

Recent research efforts have been aimed at detecting and preventing the effects of profile injection attacks. Chirita, Nejdl, and Zamfir [4] proposed several metrics for analyzing rating patterns of malicious users and introduced an algorithm specifically for detecting such attacks. Su, Zeng, and Chen [14] developed a spreading similarity algorithm in order to detect groups of very similar attackers which they applied to a simplified attack scenario. O’Mahony, Hurley and Silvestre [11] developed several techniques to defend against the attacks described in [7] and [10], including new strategies for neighborhood selection and similarity weight transformations. Our work has focused on developing a multi-strategy approach to attack detection, including supervised and unsupervised classification approaches and incorporating time-series analysis, vulnerability analysis, anomaly detection and pattern recognition. In [2] a model-based approach to detection attribute generation was introduced and shown to be effective at detecting and reducing the effects of random and average attack models. A second model-based approach for detecting attacks that target groups of items was introduced in [9] and shown to effectively detect the segment attack.

Prior work has focused on detection of the attack profiles that are reverse-engineered to introduce the largest bias in favor of the attacker. These approaches assume that attack profiles have signatures that closely resemble well-known attack models. However, an attacker may expect (and our research has shown) that hewing too closely to these optimized attack models makes the attack easier to detect [2, 9]. With detection and response schemes becoming more effective, one likely consequence is that attackers may attempt to conceal their injected attack profiles so that they more effectively masquerade as genuine profiles, while still biasing the system. We have termed such attacks *obfuscated attacks*.

The primary contribution of this work is an analysis of methods attackers may use to avoid detection schemes based on attack pattern recognition and of approaches that may limit their effectiveness. Three techniques an attacker may employ to obfuscate their attack in order to avoid detection are examined: *User shifting*, designed to reduce the similarity between profiles of an attack; *Noise injecting*, designed to blur the signature of common attack models; and *Target shifting*, designed to reduce the extreme ratings of attack profiles. We evaluate the threat of these obfuscation techniques by analyzing their success: at avoiding detection, biasing an unprotected system, and biasing a system with detection. To evaluate the effectiveness of these techniques at obfuscating an attack, we use several existing detection attributes designed to identify attacks based on the average and random attack models. We show that these obfuscated variations can introduce nearly as much bias as the original models on a system without detection, and some obfuscated attacks are more effective than the original attacks on a system with detection. We conclude by showing that the most problematic cases (those of low profile size) can be effectively handled by combining detection with a variant of the recommendation algorithm.

## 2 Profile Injection Attacks

For our purposes, a profile injection attack against a recommender system consists of a set of *attack profiles* inserted into the system with the aim of altering the system’s recommendation behavior with respect to a single target item  $i_t$ . An attack that aims to promote  $i_t$ , making it recommended more often, is called a *push attack*, and one designed to make  $i_t$  recommended less often is a *nuke attack* [10].

An *attack model* is an approach to constructing the attack profiles, based on knowledge about the recommender system’s, rating database, products, and/or users. The attack profile consists of an  $m$ -dimensional vector of ratings, where  $m$  is the total number of items in the system. The profile is partitioned in four parts as depicted in Figure 1. The null partition,  $I_\emptyset$ , are those items with no ratings in the profile. The single target item  $i_t$  will be given a rating designed

<sup>1</sup> This research was supported in part by the National Science Foundation Cyber Trust program under Grant IIS-0430303.

<sup>2</sup> Center for Web Intelligence School of Computer Science, Telecommunication and Information Systems DePaul University, Chicago, Illinois (cwilli43, mobasher, rburke, jsandvig, rbhaumik)@cs.depaul.edu

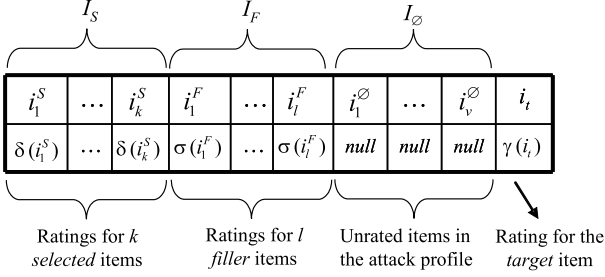


Figure 1. The general form of an attack profile.

to bias its recommendations, generally this will be either the maximum or minimum possible rating, depending on the attack type. As described below, some attacks require identifying a group of items for special treatment during the attack. This special set  $I_S$  usually receives high ratings to make the profiles similar to those of users who prefer these product. Finally, there is a set of filler items  $I_F$  whose ratings are added to complete the profile. It is the strategy for selecting items in  $I_S$  and  $I_F$  and the ratings given to these items that define an attack model and give it its character.

## 2.1 Standard Attack Models

Two basic attack models, originally introduced in [7] are *random attack*, and *average attack*. In our formalism, for these two basic attacks  $I_S$  is empty, and the contents of  $I_F$  are selected randomly. For random attack all items in  $I_F$  are assigned ratings based on the function  $\sigma$ , which generates random ratings centered around the overall average rating in the database. The average attack is very similar, but the rating for each filler item in  $I_F$  is computed based on more specific knowledge of the individual mean for each item. For more complex attacks the  $I_S$  set may be used to leverage additional knowledge about a set of items. For example the bandwagon attack [3] selects a number of popular movies for the  $I_S$  set which are given high ratings and  $I_F$  is populated as described in the random attack. The segment attack [3] populates  $I_S$  with a number of related items which it gives high ratings to as specified by  $\delta$ , while the  $I_F$  partition is given the minimum rating in order to target a segment of users. Due to space limitations we focus on the average and random attack models as they are the most widely discussed.

## 2.2 Obfuscated Attack Models

Attacks that closely follow one of the models mentioned above can be detected and their impact can be significantly reduced [2, 9]. As a result, to significantly bias the system an attacker would need to deviate from these known models to avoid detection. To explore this problem, we have examined three ways existing attack models might be obfuscated to make their detection more difficult: *noise injection*, *user shifting* and *target shifting*.

**Noise Injection** – involves adding a Gaussian distributed random number multiplied by  $\alpha$  to each rating within a set of attack profile items  $O_{ni}$ ; where  $O_{ni}$  is any subset of  $I_F \cup I_S$  to be obfuscated and  $\alpha$  is a constant multiplier governing the amount of noise to be added. This noise can be used to blur the profile signatures that are often associated with known attack models. For example, the abnormally high correlation that is associated with profiles of an average attack could be reduced by this technique while still maintaining a strong

enough similarity between the profiles and real system users.

**User Shifting** – involves incrementing or decrementing (shifting) all ratings for a subset of items per attack profile in order to reduce the similarity between attack users. More formally, for all items  $i$  in  $O_s$ ,  $r'_{i,u} = r_{i,u} + \text{shift}(u, O_s)$  where  $O_s$  is any subset of  $I_F \cup I_S$  to be obfuscated,  $r_{i,u}$  is the original assigned rating given to item  $i$  by attack profile  $u$ ,  $r'_{i,u}$  is the rating assigned to item  $i$  by the obfuscated attack profile  $u$ , and  $\text{shift}(u, O_s)$  is a function governing the amount to either increment or decrement all ratings within set  $O_s$  for profile  $u$ . This technique results in a portion of the base attack model ratings deviating for each attack profile. As a result, the distribution signature for the profile can deviate from the profile signature usually associated with the base attack model. This technique can also be used to reduce the similarity between attack profiles that often occurs in the reverse-engineered attack models.

**Target Shifting** – for a push attack is simply shifting the rating given to the target item from the maximum rating to a rating one step lower, or in the case of nuke attacks increasing the target rating to one step above the lowest rating. Although a minor change, this has a key effect. Since all reverse-engineered models dictate giving the target item the highest or lowest rating, any profile that does not include these ratings is likely to be less suspect. Naturally, profiles that are not as extreme in their preference will generate less bias in the attacked system (and our experiments bear this out). However, in many practical scenarios, for example, trying to push an item with low ratings, a target shifted attack may be almost as effective as an ordinary one.

While there are numerous ways a profile may be constructed to avoid detection, we focus on these to illustrate the detection challenges that can occur with even minor changes to existing models.

## 3 Detecting Attack Profiles

We outline a set of detection attributes that have been introduced for detecting attacks based on supervised learning techniques. Our goal is to apply the attributes introduced in [4, 2, 9] and some additional attributes to see their effectiveness at detecting obfuscated attacks when trained on base attacks. For this method, training data is created by combining genuine profiles from historic data with attack profiles inserted following the base attack models described above. Each profile is labeled as either an attack or as a genuine user. A binary classifier is then created based on this set of training data using the attributes described below and any profile classified as an attack will not be used in predictions.

These attributes come in two varieties: generic and model-specific. The generic attributes are modeled on basic descriptive statistics that attempt to capture some of the characteristics that will tend to differentiate an attacker's profile from a genuine user. The model-specific attributes attempt to detect characteristics of specific attack models.

### 3.1 Generic Attributes

Generic attributes are based on the hypothesis that the overall statistical signature of attack profiles will differ from that of authentic profiles. This difference comes from two sources: the rating given the target item, and the distribution of ratings among the filler items. As many researchers have theorized [7, 4, 10, 8], it is unlikely if not unrealistic for an attacker to have complete knowledge of the ratings

in a real system. As a result, generated profiles are likely to deviate from rating patterns seen for authentic users.

For the detection classifier’s data set we have used a number of generic attributes to capture these distribution differences. These attributes are:

- *Rating Deviation from Mean Agreement* [4], is intended to identify attackers through examining the profile’s average deviation per item, weighted by the inverse of the number of ratings for that item.
- *Weighted Degree of Agreement* [9], captures the sum of the differences of the profile’s ratings from the item’s average rating divided by the item’s rating frequency.
- *Weighted Deviation from Mean Agreement* (WDMA), designed to help identify anomalies, places a high weight on rating deviations for sparse items. We have found it to provide the highest information gain for obfuscated attacks as well as standard attacks of the attributes we have studied. The WDMA attribute can be computed in the following way:

$$WDMA_u = \frac{\sum_{i=0}^{n_u} \frac{|r_{u,i} - \bar{r}_i|}{l_i^2}}{n_u}$$

where  $U$  is the universe of all users  $u$ ; let  $P_u$  be a profile for user  $u$ , consisting of a set of ratings  $r_{u,i}$  for some items  $i$  in the universe of items to be rated; let  $n_u$  be the size of this profile in terms of the numbers of ratings; and let  $l_i$  be the number of ratings provided for item  $i$  by all users, and  $\bar{r}_i$  be the average of these ratings.

- *Degree of Similarity with Top Neighbors* (DegSim) [4], captures the average similarity of a profile’s  $k$  nearest neighbors. As researchers have hypothesized attack profiles are likely to have a higher similarity with their top 25 closest neighbors than real users [4, 12]. We also include a second slightly different attribute *DegSim'*, which captures the same metric as DegSim, but is based on the average similarity discounted if the neighbor shares fewer than  $d$  ratings in common. We have found this variant provides higher information gain at low filler sizes. The User Shifting obfuscation technique is specifically designed to reduce the effectiveness of these attributes.
- *Length Variance* (LengthVar) [2] indicates how much the length of a given profile varies from the average length in the database. This attribute is particularly effective at detecting large profile sizes often associated with bots as few authentic users have rated as many items as these larger filler sizes require.

### 3.2 Model-Specific Attributes

In our experiments, we have found that the generic attributes are insufficient for distinguishing a attack profiles from eccentric but authentic profiles. This is especially true when the profiles are small, containing few filler items. As shown in Section 2, attacks can be characterized based on the characteristics of their partitions  $i_t$  (the target item),  $I_S$  (selected items), and  $I_F$  (filler items). Model-specific attributes are those that aim to recognize the distinctive signature of a particular attack model.

Our detection model discovers partitions of each profile that maximize its similarity to the attack model. To model this partitioning, each profile is split into two sets. The set  $P_{u,T}$  contains all items in the profile with the profile’s maximum rating (or minimum in the case of a nuke attack); the set  $P_{u,F}$  consists of all other ratings in the profile. Thus the intention is for  $P_{u,T}$  to approximate  $\{i_t\} \cup I_S$  and  $P_{u,F}$  to approximate  $I_F$ . (We do not attempt to differentiate  $i_t$  from  $I_S$ .) As these attributes are dependent on the accuracy of se-

lecting these partitions, the Target Shifting obfuscation technique is designed to reduce their partitioning accuracy.

The average attack model divides the profile into three partitions:  $i_t$  given an extreme rating,  $P_{u,F}$  given filler ratings, and unrated items. The model essentially just needs to select an item to be  $i_t$  and all other rated items become  $P_{u,F}$ . By the definition of the average attack, the filler ratings will be populated such that they closely match the rating average for each filler item. We would expect that a profile generated by an average attack would exhibit a high degree of similarity (low variance) between its ratings and the average ratings for each item except for the single item chosen as the target.

The intuition of this hypothesis is implemented by iterate through all the highly-rated items, selecting each in turn as the possible target, and then computing the mean variance between the non-target (filler) items and the average across all users. Where this metric is minimum, the target item is the one most compatible with the hypothesis of the profile as being generated by an average attack, and the magnitude of the variance is an indicator of how confident we might be with this hypothesis. The Noise Injection obfuscation technique is designed to increase the filler variance thus making the profile more resemble an authentic profile, while still being very similar to the consensus item rating. The partitioning is performed twice, once for a push attack as described above and once for a nuke attack selecting low-rated items as hypothesized targets. These two partitioning sets are used to create two sets of the following attributes introduced in [2]:

- *Filler Mean Variance*, the partitioning metric described above.
- *Filler Mean Difference*, which is the average of the absolute value of the difference between the user’s rating and the mean rating (rather than the squared value as in the variance.)
- *Profile Variance*, capturing within-profile variance as this tends to be low compared to authentic users

The group attack detection model was designed for detecting attacks that target a group of items such as the Segment and Bandwagon attack, however it is included here as it has been found to be informative for single target attacks as well [9]. For this detection model, the partitioning feature that maximizes the attack’s effectiveness is the difference in ratings of items in the  $P_{u,T}$  set compared to the items in  $P_{u,F}$  captured as the *Filler Mean Target Difference* (FMTD) attribute. The effectiveness of this attribute is also reduced by the Target Shifting obfuscation technique as the difference between the filler items and target item is decreased.

All of the attributes thus far have concentrated on inter-profile statistics; target focus, however, concentrates on intra-profile statistics. The goal is to use the fact that an attacker often must introduce more than a single profile in order to achieve their desired bias. It is therefore profitable to examine the density of target items across profiles. One of the advantages of the partitioning associated with the model-based attributes described above is that a set of suspected targets is identified for each profile. For our *Target Model Focus* attribute (TMF), we calculate the degree to which the partitioning of a given profile focuses on items common to other attack partitions, and therefore measures a consensus of suspicion regarding each profile. Thus from an obfuscation perspective, if the techniques described above can reduce the accuracy of the targets selected by the above models, the effectiveness of this attribute will be reduced as well.

## 4 Experimental Methodology

**Recommendation Algorithm** – The standard  $k$ NN collaborative filtering algorithm is based on user-to-user similarity [5]. In selecting neighbors, we have used Pearson’s correlation coefficient for similar-

ities and a neighborhood size  $k = 20$ . Neighbors with a similarity of less than 0.1 are filtered out to prevent predictions from being based on distant or negative correlations. Once the most similar users are identified, predictions are calculated as described in [8]. The attack classification is incorporated by examining each profile and assigning a classification of either *attack* or *authentic*. If a profile is classified as *attack*, the profile is not used in any prediction calculations.

**Evaluation Metrics** – There has been considerable research in the area of recommender systems evaluation [6]. In evaluating security, we are interested in change in performance induced by an attack rather than raw performance. In the experiments reported below, we follow the lead of [10] in measuring an algorithms stability via prediction shift. The prediction shift metric as computed in [1] measures the change in the predicted rating of an item before and after attack.

For measuring classification performance, we use the standard measurements of precision and recall. Since we are primarily interested in how well the classification algorithms detect attack, we look at each of these metrics with respect to attack identification. Thus precision is calculated as the fraction of true positives (actual attacks) among all those profiles labeled as possible attacks, and recall is defined as the fraction of detected attacks among all of the attack profiles injected.

**Experimental Setup** – For our detection experiments, we have used the publicly-available Movie-Lens 100K dataset<sup>3</sup>. This dataset consists of 100,000 ratings on 1682 movies by 943 users. All ratings are integer values between one and five where one is the lowest (disliked) and five is the highest (most liked). Our data includes all the users who have rated at least 20 movies.

The attack detection and response experiments were conducted using a separate training and test set by partitioning the ratings data in half. The first half was used to create training data for the attack detection classifier used in later experiments. For each test the 2nd half of the data was injected with attack profiles and then run through the classifier that had been built on the augmented first half of the data. This approach was used since a typical cross-validation approach would be overly biased as the same movie being attacked would also be the movie being trained for.

For these experiments we have used 15 total detection attributes: 6 generic attributes (WDMA, RDMA, WDA, Length Variance, DegSim  $k = 450$ , and DegSim  $k = 2$  with co-rating discounting  $d = 963$ ); 6 average attack model attributes (3 for push, 3 for nuke – Filler Mean Variance, Filler Mean Difference, Profile Variance); 2 group attack model attributes (1 for push, 1 for nuke – FMTD); 1 target detection model attribute (TMF).

The training data was created using the same technique described in [9] by inserting a mix of random, average, segment, and bandwagon attacks using base attack models described above and in [9] for both push and nuke attacks at various filler sizes that ranged from 3% to 100%. Based on this training data,  $k$ NN with  $k = 9$  was used to make a binary profile classifier. To classify unseen profiles, the  $k$  nearest neighbors in the training set are used to determine the class using one over Pearson correlation distance weighting. Classification results and the  $k$ NN classifier were created using Weka [13].

In all experiments, to ensure the generality of the results, 50 movies were selected at random representing a wide range of average ratings and number of ratings. Each of these movies were attacked individually and the average is reported for all experiments. The experiments used a sample of 50 users mirroring the overall distribution of users in terms of number of movies seen and ratings provided. The

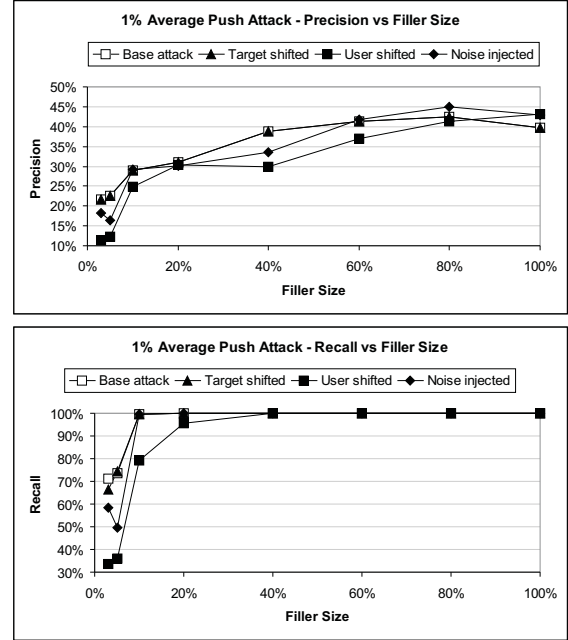


Figure 2. Classification results for 1% Average push attack.

results reported below represent averages over the combinations of test users and test movies.

To evaluate the obfuscation methods discussed above we have examined these techniques on the average and random attack models for both push and nuke attacks. For the user shift technique, for both models we shifted all of the filler items, and we used a Gaussian distributed random number for shift amount. For the noise injection technique we add noise to all of the filler items using a Gaussian distributed random number multiplied by 0.2.

## 5 Experimental Results and Discussion

In our first set of experiments we compare the attack detection model’s ability to detect the obfuscated attacks compared to the base attacks (standard non-obfuscated attacks). As Figure 2 depicts the target shifting obfuscation has little impact on the detection of average attack. The user shifted and noise injection techniques were much more successful particularly at lower filler sizes where the recall degraded over 37% for average attack. (Results for the random attack were similar.) Thus as the number of ratings increase, the patterns that distinguish an attacker would become more apparent. The same trends emerged for both average and random nuke attacks (results omitted). Recall of the nuke average attack dropped by over 30% for user shifting and noise injection, while recall of random attack degraded by over 50%. Once again target shifting alone was not particularly effective at disguising either of these attacks. Target shifting may be more significant for models such as segment attack since attributes designed to detect these attacks focus on target/filler rating separation [9]. We intend to investigate obfuscating these types of attacks in future work.

We also examined the impact on prediction shift due to deviating from the reverse-engineered attacks to avoid detection. We compared the prediction shift of base attacks and obfuscated attacks on a system without detection. Figure 3 depicts the maximum prediction shift found for each attack across all filler sizes with the black

<sup>3</sup> <http://www.cs.umn.edu/research/GroupLens/data/>

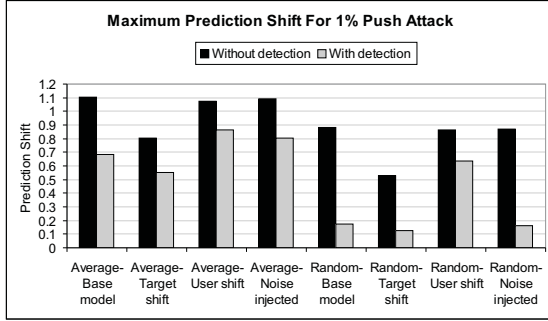


Figure 3. Maximum prediction shift 1% push attacks across all filler sizes.

bars capturing the results against a system without detection and the gray bars the results against a system with detection (we used filler sizes between 3% and 10%). As the results show, the user-shifted and noise-injected versions are nearly as effective as the non-obfuscated versions without detection for both attack models at their most effective filler sizes. This means an attacker can mount an effective attack using the obfuscation techniques with reduced chance of detection.

For both average and random attacks the user shifting obfuscation is the most effective against a system that uses detection as seen in the gray bars in Figure 3. Noise injection, however, is more effective than the base attack against a system with detection for average attack, but the obfuscated version is slightly less effective for random attack. Intuitively, this makes sense since the random attack already is an attack based on noise, and its lack of correlation to item averages is one of the features that aides in its detection; additional noise being added is unlikely to improve the correlation.

The classification and prediction shift results indicate that, when combined with detection, average and random attacks at lower filler sizes pose the most risk. To reduce the effect of these attacks at lower filler sizes, one approach would be to discount profiles that have fewer items in their profile. Herlocker *et al.* introduced such a variation in [5] that discounts similarity between profiles that have fewer than 50 co-rated items by  $n/50$  where  $n$  is the number of co-rated items. While this modification was proposed originally to improve prediction quality, it has some interesting effects on changing the characteristics of effective attacks as well. As Figure 4 shows, while the average and random attacks are about as effective against the co-rate discounted version as they are against the basic version at high filler sizes, at low filler sizes their impact is far less. When combined with the detection model outlined above the largest prediction shift achieved by any of the attacks described above is only .06 compared to the .86 shift achieved against basic  $k$ NN. This combination may not be as effective against attacks that focus specifically on popular items, since they are designed to increase the likelihood of co-rating, but it does appear to add significant robustness for the attacks studied in this paper.

A more challenging problem will likely be ensuring robustness against unknown attacks as profile classification alone may be insufficient. Unlike traditional classification problems where patterns are observed and learned, in this context there is a competitive aspect since attackers are motivated to actively look for ways to beat the classifier. Given this dynamic, a solution may lie in combining multiple detection approaches such as time series or rating distribution analysis. We envision combining the techniques above with other detection techniques to create a comprehensive detection framework.

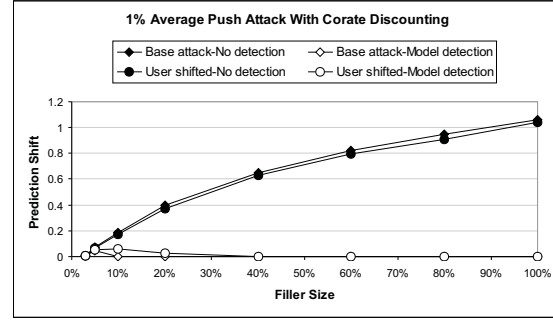


Figure 4. Prediction shift against co-rate discounted  $k$ NN.

## REFERENCES

- [1] R. Burke, B. Mobasher, and R. Bhaumik, 'Limited knowledge shilling attacks in collaborative filtering systems', in *Proceedings of the 3rd IJ-CAI Workshop in Intelligent Techniques for Personalization*, Edinburgh, Scotland, (August 2005).
- [2] R. Burke, B. Mobasher, C. Williams, and R. Bhaumik, 'Detecting profile injection attacks in collaborative recommender systems', in *To appear in Proceedings of the IEEE Joint Conference on E-Commerce Technology and Enterprise Computing, E-Commerce and E-Services (CEC/EEE 2006)*, Palo Alto, CA, (June 2006).
- [3] R. Burke, B. Mobasher, R. Zabicki, and R. Bhaumik, 'Identifying attack models for secure recommendation', in *Beyond Personalization: A Workshop on the Next Generation of Recommender Systems*, San Diego, California, (January 2005).
- [4] Paul-Alexandru Chirita, Wolfgang Nejdl, and Cristian Zamfir, 'Preventing shilling attacks in online recommender systems', in *WIDM '05: Proceedings of the 7th annual ACM international workshop on Web information and data management*, pp. 67–74, New York, NY, USA, (2005). ACM Press.
- [5] J. Herlocker, J. Konstan, A. Borchers, and J. Riedl, 'An algorithmic framework for performing collaborative filtering', in *Proceedings of the 22nd ACM Conference on Research and Development in Information Retrieval (SIGIR'99)*, Berkeley, CA, (August 1999).
- [6] J. Herlocker, J. Konstan, L. G. Tervin, and J. Riedl, 'Evaluating collaborative filtering recommender systems', *ACM Transactions on Information Systems*, **22**(1), 5–53, (2004).
- [7] S. Lam and J. Reidl, 'Shilling recommender systems for fun and profit', in *Proceedings of the 13th International WWW Conference*, New York, (May 2004).
- [8] B. Mobasher, R. Burke, R. Bhaumik, and C. Williams, 'Effective attack models for shilling item-based collaborative filtering systems', in *Proceedings of the 2005 WebKDD Workshop, held in conjunction with ACM SIGKDD '05*, Chicago, Illinois, (August 2005).
- [9] B. Mobasher, R. Burke, C. Williams, and R. Bhaumik, 'Analysis and detection of segment-focused attacks against collaborative recommendation', in *To appear in Lecture Notes in Computer Science: Proceedings of the 2005 WebKDD Workshop*. Springer, (2006).
- [10] M. O'Mahony, N. Hurley, N. Kushmerick, and G. Silvestre, 'Collaborative recommendation: A robustness analysis', *ACM Transactions on Internet Technology*, **4**(4), 344–377, (2004).
- [11] M.P. O'Mahony, N.J. Hurley, and G. Silvestre, 'Utility-based neighbourhood formation for efficient and robust collaborative filtering', in *Proceedings of the 5th ACM Conference on Electronic Commerce (EC04)*, pp. 260–261, (May 2004).
- [12] Paul Resnick, Neophytos Iacovou, Mitesh Suchak, Peter Bergstrom, and John Riedl, 'GroupLens: an open architecture for collaborative filtering of netnews', in *CSCW '94: Proceedings of the 1994 ACM conference on Computer supported cooperative work*, pp. 175–186. ACM Press, (1994).
- [13] I. H. Witten and E. Frank, *Data Mining: Practical machine learning tools and techniques, 2nd Edition*, Morgan Kaufmann, San Francisco, CA, 2005.
- [14] Hua-Jun Zeng, Xue-Feng Su and Z. Chen., 'Finding group shilling in recommendation system', in *WWW 05 Proceedings of the 14th international conference on World Wide Web*, (May 2005).